BIT Numerical Mathematics (2024) 64:4
https://doi.org/10.1007/s10543-023-01004-7

BIT

Incremental algorithms for truncated higher-order singular value decompositions

Chao Zeng[1] · **Michael K. Ng**[2] · **Tai-Xiang Jiang**[3]

Received: 11 March 2023 / Accepted: 3 December 2023

Abstract

We develop and study incremental algorithms for truncated higher-order singular value decompositions. By combining the SVD updating and different truncated higher-order singular value decompositions, two incremental algorithms are proposed. Not only the factor matrices but also the core tensor are updated in an incremental style. The costs of these algorithms are compared and the approximation errors are analyzed. Numerical results demonstrate that the proposed incremental algorithms have advantages in online computation.

Keywords SVD updating · Incremental algorithm · Truncated higher-order singular value decomposition

Communicated by Rosemary Anne Renaut.

Chao Zeng's work was supported in part by the National Natural Science Foundation of China (12201319) and the Fundamental Research Funds for the Central Universities, Nankai University (63231142). Michael K. Ng's research supported in part by the HKRGC GRF 17201020 and 17300021, and CRF C7004-21GF, and Joint NSFC and RGC NHKU769/21. Tai-Xiang Jiang's work was supported in part by the National Natural Science Foundation of China (12001446), Natural Science Foundation of Sichuan, China (2022NSFSC1798), the Fundamental Research Funds for the Central Universities, and the Guanghua Talent Project.

✉ Tai-Xiang Jiang
taixiangjiang@gmail.com

Chao Zeng
zengchao@nankai.edu.cn

Michael K. Ng
michael-ng@hkbu.edu.hk

1 School of Mathematical Sciences and LPMC, Nankai University, Tianjin, People's Republic of China

2 Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong

3 School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, Sichuan, People's Republic of China

Published online: 08 January 2024

Mathematics Subject Classification 15A69 · 15A72 · 65F99 · 65Y20

1 Introduction

Tensors are multidimensional arrays and used to represent ubiquitous high-dimensional data, such as color images, hyperspectral images, videos, and facial recognition datasets. Like singular value decomposition (SVD) for matrix analysis, tensor decompositions are basic tools for high-dimensional data analysis. Applications include data compression, feature extraction, knowledge discovery, and more. Various decompositions such as CANDECOMP/PARAFAC [4, 14], higher-order singular value decomposition (HOSVD) [7, 34], hierarchical SVD [10, 13], T-SVD [16, 17], tensor-train [24] have been developed in the literature. The interested reader can refer to [18] for a thorough review.

One of the most widely used tensor decompositions is HOSVD. Although the HOSVD is regarded as a convincing multilinear generalization of the SVD, there are some differences between them. First, the right singular vectors of unfolding matrices are not involved in the HOSVD, and the core tensor corresponding to the HOSVD is computed by matrix-tensor products. Second, the best low-rank approximation of a matrix is just the truncated SVD, while there is no closed-form optimal solution for the low-multilinear-rank approximation of a tensor [18, 37], resulting in multiple definitions of the truncated HOSVD. Two main types of the truncated HOSVD are T-HOSVD and ST-HOSVD; see Sect. 2.3 for details. Recently, randomized algorithms have been developed for the truncated HOSVD; see [21, 33]. These methods are suitable for low-multilinear-rank tensors.

In many applications, we need to handle tensors with dynamic sizes. One situation is that only partial data sets are known at first and the new data sets are arriving continuously over time or are available later. Examples include live video streams, surveillance videos, and driving recorder datasets. Such tensors are called tensor streams or incremental tensors; see [31, 32] for instance. To tackle an incremental tensor, we usually need to compute the decomposition once a new data set arrives. For example, in online object tracking [15, 27], we need to give the tracking result once a new frame arrives. Most algorithms [7, 18, 37] of tensor decompositions are designed for the batch mode. Batch-mode algorithms, which need to compute the decomposition from scratch when new data set arrives, are not suitable for online computation because of their poor performance in terms of time and space cost for large-scale incremental tensors. For incremental tensors, developing online algorithms to make use of the known decomposition of the old tensor and reduce the computation cost becomes necessary.

For the matrix case, there are two main types of online algorithms for SVD: downdating and updating. Downdating [12] is to find the SVD of a matrix obtained by deleting a column from the original matrix, while updating is to find the SVD of a matrix obtained by adding a column to the original matrix. There are a lot of works on the SVD updating, e.g., [2, 3, 11, 22, 26]. In this work, we will develop online algorithms for incremental tensors, which can be regarded as extensions of the SVD updating. Meanwhile, we focus on the incremental HOSVD, and incremental tensor

CANDECOMP/PARAFAC decompositions can be found in [19, 23, 25, 29, 36, 38, 39].

The incremental HOSVD has been widely studied in [5, 15, 20, 27, 30, 35].[1] However, All existing works on incremental HOSVD do not study this problem systematically. In [5, 15, 27], only the update of factor matrices is discussed. In [20, 30, 35], the factor matrices are updated by incremental algorithms, but the core tensor is updated simply by matrix-tensor products, which is a batch mode computation; see Remark 2 for details. Thus the cost (space and time) can be very large and the algorithms are not suitable for long time computation. All these works are aimed at incremental algorithms for the T-HOSVD, while incremental algorithms for the ST-HOSVD have not been studied in the literature. Moreover, the approximation errors of the incremental HOSVD have not been discussed before.

In this paper, we discuss incremental algorithms for the truncated HOSVD. Here we assume that the new data are coming in a specific mode (for example, we are interested in the time dimension of video data) and the truncation rank is fixed in the incremental process. In summary, the main contributions of this paper are listed as follows:

1. We propose an incremental algorithm corresponding to the T-HOSVD. The factor matrices and the core tensor are all updated by incremental methods.
2. We propose an incremental algorithm corresponding to the ST-HOSVD.
3. The costs of different methods are compared and the approximation errors are analyzed.

This paper is organized as follows. Section 2 introduces basic definitions about tensors. The incremental algorithms are proposed in Sect. 3. Bounds of the approximation errors are established in Sect. 4. Numerical experiments are provided in Sect. 5. This paper is ended with conclusions in Sect. 6.

2 Preliminaries

2.1 Notation and definitions

We use bold-face lowercase letters ($\mathbf{a}, \mathbf{b}, \ldots$) to denote vectors, bold-face capitals ($\mathbf{A}, \mathbf{B}, \ldots$) to denote matrices and calligraphic letters ($\mathcal{A}, \mathcal{B}, \ldots$) to denote tensors. The notations $\mathbf{I}$ and $\mathbf{0}$ denote the identity matrix and the zero matrix of suitable dimensions. The Frobenius norm of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is given by

$$\|\mathcal{A}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \cdots i_N}^2},$$

where $a_{i_1 i_2 \cdots i_N}$ is the $(i_1, i_2, \ldots, i_N)$th entry of $\mathcal{A}$.

[1] Although the algorithm proposed in [33] is aimed at streaming tensors, it forms an approximation of the tensor only after all the data have been observed and hence is not an incremental algorithm.

Given $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N}$, $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times J_N}$, we denote by $\mathcal{C} := \begin{bmatrix} \mathcal{A} & \mathcal{B} \end{bmatrix}$ an $I_1 \times \cdots \times I_{N-1} \times (I_N + J_N)$ tensor satisfying

$$\mathcal{C}(:, \ldots, :, 1 : I_N) = \mathcal{A}, \quad \mathcal{C}(:, \ldots, :, I_N + 1 : I_N + J_N) = \mathcal{B}.$$

The *mode-n fibers* of a tensor are the higher-order analogue of matrix column and row vectors. A tensor can be vectorized into a vector with specific ordering. We adopt the following simple ordering in Matlab:

$$\mathrm{vec}(\mathcal{A}) := \mathcal{A}(:).$$

A tensor can also be unfolded into a matrix. The mode-n *unfolding matrix* of $\mathcal{A}$ is denoted by $\mathbf{A}_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. The specific ordering of the mode-n vectors within this unfolding is usually not important, as long as it is consistent. We adopt the canonical ordering, as presented in [18].

The n-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by a matrix $\mathbf{M}$, denoted by $\mathbf{M} \cdot_n \mathcal{A}$, is a tensor generated by multiplying each mode-n fiber of $\mathcal{A}$ by $\mathbf{M}$. If $n > N$, $\mathcal{A}$ is regarded as an nth-order tensor with size $I_1 \times \cdots \times I_N \times 1 \times \cdots \times 1$. Following [9], we write $\mathbf{M}_1 \cdot_1 \cdots \mathbf{M}_n \cdot_n \mathcal{A}$ more concisely as $(\mathbf{M}_1, \ldots, \mathbf{M}_n) \cdot \mathcal{A}$,[2] which is the called the *multilinear multiplication* of $\mathcal{A}$ by $(\mathbf{M}_1, \ldots, \mathbf{M}_n)$.

The n-rank of a tensor $\mathcal{A}$, denoted by $r_n(\mathcal{A})$ is the dimension of the vector space spanned by all mode-n fibers. The N-tuple $(r_1(\mathcal{A}), \ldots, r_N(\mathcal{A}))$ is called the *multilinear rank* of $\mathcal{A}$.

2.2 Incremental SVD

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, suppose its SVD is

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{m \times r}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$, $\mathbf{V} \in \mathbb{R}^{n \times r}$. By appending a new row $\mathbf{b}^\top \in \mathbb{R}^{1 \times n}$, we obtain a new matrix $\mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{bmatrix}$. We want to obtain the SVD of $\mathbf{B}$ based on the SVD of $\mathbf{A}$. The resulting decomposition is called the incremental SVD of $\mathbf{B}$.

We have

$$\mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{bmatrix} = \begin{bmatrix} \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \\ \mathbf{b}^\top \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \\ & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma} & \\ & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}^\top \\ \mathbf{b}^\top \end{bmatrix}. \tag{1}$$

Consider the Gram-Schmidt orthonormalization of $\mathbf{b}$ with respect to $\mathbf{V}$:

$$\mathbf{x} = \mathbf{V}^\top \mathbf{b}, \quad \mathbf{y} = \mathbf{b} - \mathbf{V}\mathbf{x}, \quad \rho = \|\mathbf{y}\|, \quad \mathbf{p} = \frac{\mathbf{y}}{\rho}.$$

[2] Some papers such as [7, 8] use $\mathcal{A} \times_1 \mathbf{M}_1 \cdots \times_n \mathbf{M}_n$ in place of $(\mathbf{M}_1, \ldots, \mathbf{M}_n) \cdot \mathcal{A}$. Here we follow the notation of [9, 37]. Refer to [9, Section 2.1] for more discussions.

Then, we have

$$\begin{bmatrix}\mathbf{V} & \mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{V} & \mathbf{p}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{x}\\ \mathbf{0} & \rho\end{bmatrix}$$

and

$$\begin{bmatrix}\boldsymbol{\Sigma} & \\ & 1\end{bmatrix}\begin{bmatrix}\mathbf{V}^\top\\ \mathbf{b}^\top\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Sigma} & \mathbf{0}\\ \mathbf{x}^\top & \rho\end{bmatrix}\begin{bmatrix}\mathbf{V}^\top\\ \mathbf{p}^\top\end{bmatrix}.$$

Define

$$\mathbf{D} := \begin{bmatrix}\boldsymbol{\Sigma} & \mathbf{0}\\ \mathbf{x}^\top & \rho\end{bmatrix} \in \mathbb{R}^{(r+1)\times(r+1)}.$$

Suppose the SVD of $\mathbf{D}$ is $\mathbf{D} = \mathbf{G}\boldsymbol{\Lambda}\mathbf{H}^\top$. Combining the SVD of $\mathbf{D}$ and (1) gives

$$\mathbf{B} = \begin{bmatrix}\mathbf{A}\\ \mathbf{b}^\top\end{bmatrix} = \left(\begin{bmatrix}\mathbf{U} & \\ & 1\end{bmatrix}\mathbf{G}\right)\boldsymbol{\Lambda}(\begin{bmatrix}\mathbf{V} & \mathbf{p}\end{bmatrix}\mathbf{H})^\top,$$

which is the SVD of $\mathbf{B}$. The decomposition $\mathbf{D} = \mathbf{G}\boldsymbol{\Lambda}\mathbf{H}^\top$ is useful in the incremental algorithms of the truncated HOSVD. We summarize the incremental SVD as Algorithm 1. The cost of Algorithm 1 is $O(nr + r^3)$.

Algorithm 1: Incremental SVD (iSVD)

function $[\mathbf{G}, \boldsymbol{\Lambda}, \mathbf{H}]$ = `iSVD` $(\boldsymbol{\Sigma}, \mathbf{V}, \mathbf{b})$
  $\mathbf{x} \leftarrow \mathbf{V}^\top\mathbf{b}$
  $\mathbf{y} \leftarrow \mathbf{b} - \mathbf{V}\mathbf{x}$
  $\rho \leftarrow \|\mathbf{y}\|$
  $\mathbf{p} \leftarrow \mathbf{y}/\rho$
  $\mathbf{D} \leftarrow \begin{bmatrix}\boldsymbol{\Sigma} & \mathbf{0}\\ \mathbf{x}^\top & \rho\end{bmatrix}$
  $[\mathbf{G}, \boldsymbol{\Lambda}, \mathbf{H}] = \mathtt{SVD}(\mathbf{D})$ ▷ `Compute the SVD of` $\mathbf{D}$
end function

The incremental algorithm for appending a new column is similar. We omit it here. In incremental algorithms of the truncated HOSVD, we will compute the incremental SVD of an unfolding matrix, but the situation is somewhat different: several columns are appended simultaneously, and only the left singular vectors are needed in the final result. Therefore, the corresponding algorithm is different from the incremental SVD. See Sect. 3 for details.

2.3 HOSVD, T-HOSVD and ST-HOSVD

The HOSVD is proposed in [7].

Theorem 1 (HOSVD) *Every tensor* $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *admits a higher-order singular value decomposition:*

$$\mathcal{A} = (\mathbf{U}_1, \ldots, \mathbf{U}_N) \cdot \mathcal{S},$$

where the factor matrix $\mathbf{U}_n \in \mathbb{R}^{I_n \times I_n}$ *is orthogonal, obtained from the SVD of the mode-n unfolding of* $\mathcal{A}$*:*

$$\mathbf{A}_{(n)} = \mathbf{U}_n \boldsymbol{\Sigma}_n \mathbf{V}_n^\top,$$

and the core tensor $\mathcal{S} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *can be obtained by*

$$\mathcal{S} = \left(\mathbf{U}_1^\top, \ldots, \mathbf{U}_N^\top\right) \cdot \mathcal{A}.$$

Truncating the HOSVD can be used for low-multilinear-rank approximation to a tensor. Suppose that we want to approximate $\mathcal{A}$ by a rank-$(R_1, \ldots, R_N)$ tensor with $R_n \leq I_n$ for all $1 \leq n \leq N$. A simple choice of the factor matrix $\bar{\mathbf{U}}_n$ of the truncated HOSVD (T-HOSVD) is obtained from a truncated SVD of the mode-n unfolding matrix:

$$\mathbf{A}_{(n)} = \mathbf{U}_n \boldsymbol{\Sigma}_n \mathbf{V}_n^\top = \begin{bmatrix} \bar{\mathbf{U}}_n & \bar{\mathbf{U}}_n' \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_n & \\ & \bar{\boldsymbol{\Sigma}}_n' \end{bmatrix} \begin{bmatrix} \bar{\mathbf{V}}_n^\top \\ \bar{\mathbf{V}}_n'^\top \end{bmatrix}, \tag{2}$$

where $\bar{\mathbf{U}}_n \in \mathbb{R}^{I_n \times R_n}$ consists of the R_n dominant left singular vectors. By [8, Theorem 4.1], the truncated core tensor is

$$\left(\bar{\mathbf{U}}_1^\top, \ldots, \bar{\mathbf{U}}_N^\top\right) \cdot \mathcal{A} =: \bar{\mathcal{S}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}.$$

Throughout this paper, T-HOSVD only refers to the truncated HOSVD defined by (2).

Another strategy for truncating the HOSVD is the sequentially truncated higher-order singular value decomposition (ST-HOSVD) [37], which is cheaper still to compute and often improves the approximation error with respect to the T-HOSVD. The factor matrix $\bar{\mathbf{U}}_n$ of the ST-HOSVD is obtained from a truncated SVD of the mode-n unfolding of a tensor generated by sequentially truncated the original tensor $\mathcal{A}$. Unlike the T-HOSVD, the ordering in which the modes are processed is relevant and leads to different approximations and time cost. We use a sequence $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_N]$, which is a permutation of $[1, \ldots, N]$, to denote the ordering in which modes are processed. Then, the factor matrices of ST-HOSVD are defined in the following manner: $\mathcal{A}^0 = \mathcal{A}$, for $n = 1, \ldots, N$,

$$\mathbf{A}_{(\pi_n)}^{n-1} = \begin{bmatrix} \bar{\mathbf{U}}_{\pi_n} & \bar{\mathbf{U}}_{\pi_n}' \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_{\pi_n} & \\ & \bar{\boldsymbol{\Sigma}}_{\pi_n}' \end{bmatrix} \begin{bmatrix} \bar{\mathbf{V}}_{\pi_n}^\top \\ \bar{\mathbf{V}}_{\pi_n}'^\top \end{bmatrix}, \quad \mathcal{A}^n = \bar{\mathbf{U}}_{\pi_n}^\top \cdot_{\pi_n} \mathcal{A}^{n-1}. \tag{3}$$

For more details, see [37, Algorithm 1].

Remark 1 How to choose a good processing ordering $\boldsymbol{\pi}$ is an open problem. In [37], the authors propose a heuristic choice of the processing ordering:

$$\boldsymbol{\pi} = [\pi_1, \ldots, \pi_N] \text{ such that } I_{\pi_1} \leq \ldots \leq I_{\pi_N}$$

under the assumption that the time complexity of computing the SVD of an $m \times n$ matrix is $O(mn \min(m, n))$. However, the actual running time of computing the SVD is not only determined by this simple time complexity. For simplicity, we set $\boldsymbol{\pi} = [1, \ldots, N]$ in the experiments presented in Sect. 5, which has a good performance in time cost among all processing orderings.

Property 1 ([37, Property 6.3]) Let $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be truncated to rank-$(R_1, \ldots, R_N)$ by the ST-HOSVD with processing ordering $[1, \ldots, N]$ (respectively, T-HOSVD). Assume the time complexity of computing the SVD of an $m \times n$ matrix is $O(mn \min(m, n))$. Then, the ST-HOSVD and T-HOSVD requires

$$O\left((I_1 + R_1 I_2 / I_1 + R_1) \prod_{n=1}^{N} I_n\right) \text{ and}$$
$$O\left(\sum_{n=1}^{N} I_1 \cdots I_N \min(I_n, I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)\right)$$

operations to compute the approximation, respectively.

2.4 Tensor norms with respect to multilinear multiplication

The n-mode product has the following relationship:

$$\mathcal{B} = \mathbf{M} \cdot_n \mathcal{A} \Leftrightarrow \mathbf{B}_{(n)} = \mathbf{M}\mathbf{A}_{(n)}.$$

If $\mathbf{M}_k$ has orthonormal columns for all $k = 1, \ldots, n$, then

$$\|(\mathbf{M}_1, \ldots, \mathbf{M}_n) \cdot \mathcal{A}\| = \|\mathcal{A}\|. \tag{4}$$

Suppose U_n is a subspace of $\mathbb{R}^{I_n}$ and the columns of $\mathbf{U}_n \in \mathbb{R}^{I_n \times R_n}$ form an orthonormal basis of U_n. Then the *multilinear orthogonal projection* [37] from the tensor space $\mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_N}$ onto the subspace $\mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_{n-1}} \times U_n \times \mathbb{R}^{I_{n+1}} \times \cdots \times \mathbb{R}^{I_N}$ is given by

$$\mathscr{P}_n[\mathbf{U}_n]\mathcal{A} := \left(\mathbf{U}_n \mathbf{U}_n^\top\right) \cdot_n \mathcal{A}.$$

The orthogonal complement of $\mathscr{P}_n[\mathbf{U}_n]$ is

$$\mathscr{P}_n^\perp[\mathbf{U}_n]\mathcal{A} := (1 - \mathscr{P}_n[\mathbf{U}_n])\,\mathcal{A} = \left(\mathbf{I} - \mathbf{U}_n \mathbf{U}_n^\top\right) \cdot_n \mathcal{A}.$$

Given $\mathbf{U}_n \in \mathbb{R}^{I_n \times R_n}$ satisfying $\mathbf{U}_n^\top \mathbf{U}_n = \mathbf{I}$ for $n = 1, \ldots, N$, the solution of the following problem

$$\min_{\mathcal{S} \in \mathbb{R}^{R_1 \times \cdots \times R_N}} \|\mathcal{A} - (\mathbf{U}_1, \ldots, \mathbf{U}_N) \cdot \mathcal{S}\|$$

is given by $\mathcal{S} = (\mathbf{U}_1^\top, \ldots, \mathbf{U}_N^\top) \cdot \mathcal{A}$, as proved in [8, Theorem 4.1]. This is just to say that

$$\|\mathcal{A} - \mathscr{P}_1[\mathbf{U}_1] \cdots \mathscr{P}_N[\mathbf{U}_N]\mathcal{A}\| = \min_{\mathcal{S} \in \mathbb{R}^{R_1 \times \cdots \times R_N}} \|\mathcal{A} - (\mathbf{U}_1, \ldots, \mathbf{U}_N) \cdot \mathcal{S}\|. \quad (5)$$

A special case of (5) is

$$\left\| \mathscr{P}_n^\perp[\mathbf{U}_n]\mathcal{A} \right\| = \min_{\mathcal{S} \in V_n} \|\mathcal{A} - \mathbf{U}_n \cdot_n \mathcal{S}\| \quad \forall n = 1, \ldots, N, \quad (6)$$

where $V_n = \mathbb{R}^{I_1 \times \cdots I_{n-1} \times R_n \times I_{n+1} \times \cdots \times I_N}$. It follows from [37, (5.1)] that

$$\|\mathcal{A} - \mathscr{P}_1[\mathbf{U}_1] \cdots \mathscr{P}_N[\mathbf{U}_N]\mathcal{A}\| \le \sqrt{\sum_{n=1}^{N} \left\| \mathscr{P}_n^\perp[\mathbf{U}_n]\mathcal{A} \right\|^2} \le \sum_{n=1}^{N} \left\| \mathscr{P}_n^\perp[\mathbf{U}_n]\mathcal{A} \right\|. \quad (7)$$

3 Incremental algorithms

In practice, the incremental algorithm will be run many times as many slices will be sequentially added. For ease of presentation, we introduce a tensor sequence $\mathcal{A}^0, \mathcal{A}^1, \ldots, \mathcal{A}^t, \mathcal{A}^{t+1}, \ldots$ satisfying

$$\mathcal{A}^{t+1} = \begin{bmatrix} \mathcal{A}^t & \mathcal{B}^t \end{bmatrix} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times (I_N + t + 1)}, \quad (8)$$

where $\mathcal{B}^t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1}}$ is a *slice*. Denote $J = \prod_{n=1}^{N-1} I_n$. For $n = 1, \ldots, N-1$, $\mathbf{A}_{(n)}^{t+1} \in \mathbb{R}^{I_n \times \frac{J(I_N+t+1)}{I_n}}$ is a column permutation of $\begin{bmatrix} \mathbf{A}_{(n)}^t & \mathbf{B}_{(n)}^t \end{bmatrix}$. We can use $\begin{bmatrix} \mathbf{A}_{(n)}^t & \mathbf{B}_{(n)}^t \end{bmatrix}$, which is a matrix generated by appending $\frac{J}{I_n}$ new columns to $\mathbf{A}_{(n)}^t$, to obtain the updated factor matrix. For $n = N$, $\mathbf{A}_{(N)}^{t+1} = \begin{bmatrix} \mathbf{A}_{(N)}^t \\ \mathrm{vec}(\mathcal{B}^t)^\top \end{bmatrix} \in \mathbb{R}^{(I_N+t+1) \times J}$ is a matrix generated by appending a new row to $\mathbf{A}_{(N)}^t$.

The truncation rank is $(R_1, \ldots, R_N)$. Throughout this section, we denote the HOSVD and truncated HOSVD of $\mathcal{A}^t$ by

$$\mathcal{A}^t = \left(\mathbf{U}_1^t, \ldots, \mathbf{U}_N^t\right) \cdot \mathcal{S}^t \approx \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t.$$

We assume that the truncated HOSVD of $\mathcal{A}^t$ is known and present two incremental algorithms for computing the truncated HOSVD of $\mathcal{A}^{t+1}$.

3.1 iT-HOSVD

We can compute the truncated factor matrix $\bar{\mathbf{U}}_n^{t+1}$ from the truncated factor matrix $\bar{\mathbf{U}}_n^t$, which is extended from the incremental SVD introduced in Sect. 2.2. We call this method *incremental truncated HOSVD* (iT-HOSVD).

First, we introduce how to update truncated factor matrices.

1. The case $n = 1, \ldots, N-1$. We have

$$\begin{bmatrix} \mathbf{A}_{(n)}^t & \mathbf{B}_{(n)}^t \end{bmatrix} \approx \begin{bmatrix} \bar{\mathbf{U}}_n^t \bar{\mathbf{\Sigma}}_n^t \bar{\mathbf{V}}_n^{t\top} & \mathbf{B}_{(n)}^t \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{B}_{(n)}^t \end{bmatrix} \begin{bmatrix} \bar{\mathbf{\Sigma}}_n^t & \\ & \mathbf{I} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{V}}_n^{t\top} & \\ & \mathbf{I} \end{bmatrix}. \quad (9)$$

Define

$$\mathbf{X}_n^t := \bar{\mathbf{U}}_n^{t\top} \mathbf{B}_{(n)}^t, \quad \mathbf{Y}_n^t := \mathbf{B}_{(n)}^t - \bar{\mathbf{U}}_n^t \mathbf{X}_n \in \mathbb{R}^{I_n \times \frac{J}{I_n}}. \quad (10)$$

Let $\mathbf{P}_n^t$ be an orthonormal basis of the column space of $\mathbf{Y}_n^t$, which can be solved by the QR decomposition. Now, we have

$$\begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{B}_{(n)}^t \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{P}_n^t \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{X}_n^t \\ \mathbf{0} & \mathbf{P}_n^{t\top} \mathbf{Y}_n^t \end{bmatrix}$$

and

$$\begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{B}_{(n)}^t \end{bmatrix} \begin{bmatrix} \bar{\mathbf{\Sigma}}_n^t & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{P}_n^t \end{bmatrix} \begin{bmatrix} \bar{\mathbf{\Sigma}}_n^t & \mathbf{X}_n^t \\ \mathbf{0} & \mathbf{P}_n^{t\top} \mathbf{Y}_n^t \end{bmatrix}.$$

Define

$$\mathbf{D}_n^t := \begin{bmatrix} \bar{\mathbf{\Sigma}}_n^t & \mathbf{X}_n^t \\ \mathbf{0} & \mathbf{P}_n^{t\top} \mathbf{Y}_n^t \end{bmatrix} \in \mathbb{R}^{I_n \times (\frac{J}{I_n} + R_n)}. \quad (11)$$

Suppose the SVD of $\mathbf{D}_n^t$ is $\mathbf{D}_n^t = \mathbf{G}_n^t \mathbf{\Lambda}_n^t \mathbf{H}_n^{t\top}$. Combining the SVD of $\mathbf{D}_n^t$ and (9) gives

$$\begin{bmatrix} \mathbf{A}_{(n)}^t & \mathbf{B}_{(n)}^t \end{bmatrix} \approx \left(\begin{bmatrix} \bar{\mathbf{U}}_n^t & \mathbf{P}_n^t \end{bmatrix} \mathbf{G}_n^t \right) \mathbf{\Lambda}_n^t \left(\begin{bmatrix} \mathbf{V}_n^t & \\ & \mathbf{I} \end{bmatrix} \mathbf{H}_n^t \right)^\top. \quad (12)$$

Define $\mathbf{\Phi}_n^t := \mathbf{G}_n^t(1 : R_n, 1 : R_n)$, $\mathbf{\Psi}_n^t := \mathbf{G}_n^t(R_n + 1 : I_n, 1 : R_n)$. Then

$$\bar{\mathbf{U}}_n^{t+1} = \bar{\mathbf{U}}_n^t \mathbf{\Phi}_n^t + \mathbf{P}_n^t \mathbf{\Psi}_n^t, \quad \bar{\mathbf{\Sigma}}_n^{t+1} = \mathbf{\Lambda}_n^t(1 : R_n, 1 : R_n). \quad (13)$$

2. The case $n = N$. This case is in the framework of Sect. 2.2. Here we write the following formula (line 13 of Algorithm 2) for the convenience of later discussion:

$$\bar{\mathbf{U}}_N^{t+1} = \begin{bmatrix} \bar{\mathbf{U}}_N^t \boldsymbol{\Phi}_N^t \\ \boldsymbol{\Psi}_N^t \end{bmatrix}. \tag{14}$$

Now we discuss how to update the truncated core tensor. Define

$$\mathbf{E}_1 := \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(I_N+t+1)\times(I_N+t)}, \quad \mathbf{E}_2 := \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \in \mathbb{R}^{(I_N+t+1)\times 1}.$$

We have

$$\mathcal{A}^{t+1} = \begin{bmatrix} \mathcal{A}^t & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathcal{B}^t \end{bmatrix} = \mathbf{E}_1 \cdot_N \mathcal{A}^t + \mathbf{E}_2 \cdot_N \mathcal{B}^t.$$

It follows from (14) that

$$\bar{\mathbf{U}}_N^{t+1^\top} \mathbf{E}_1 = \begin{bmatrix} \boldsymbol{\Phi}_N^{t^\top} \bar{\mathbf{U}}_N^{t^\top} & \boldsymbol{\Psi}_N^{t^\top} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} = \boldsymbol{\Phi}_N^{t^\top} \bar{\mathbf{U}}_N^{t^\top}, \quad \bar{\mathbf{U}}_N^{t+1^\top} \mathbf{E}_2 = \boldsymbol{\Psi}_N^{t^\top}.$$

Define $\mathcal{T}^t := \bar{\mathbf{U}}_N^{t^\top} \cdot_N \mathcal{A}^t$. Then

$$\begin{aligned} \mathcal{T}^{t+1} &= \bar{\mathbf{U}}_N^{t+1^\top} \cdot_N \mathcal{A}^{t+1} = \bar{\mathbf{U}}_N^{t+1^\top} \cdot_N \left(\mathbf{E}_1 \cdot_N \mathcal{A}^t + \mathbf{E}_2 \cdot_N \mathcal{B}^t \right) \\ &= \boldsymbol{\Phi}_N^{t^\top} \cdot_N \left(\bar{\mathbf{U}}_N^{t^\top} \cdot_N \mathcal{A}^t \right) + \boldsymbol{\Psi}_N^{t^\top} \cdot_N \mathcal{B}^t = \boldsymbol{\Phi}_N^{t^\top} \cdot_N \mathcal{T}^t + \boldsymbol{\Psi}_N^{t^\top} \cdot_N \mathcal{B}^t. \end{aligned} \tag{15}$$

and

$$\bar{\mathcal{S}}^{t+1} = \left(\bar{\mathbf{U}}_1^{t+1^\top}, \ldots, \bar{\mathbf{U}}_N^{t+1^\top} \right) \cdot \mathcal{A}^{t+1} = \left(\bar{\mathbf{U}}_1^{t+1^\top}, \ldots, \bar{\mathbf{U}}_{N-1}^{t+1^\top} \right) \cdot \mathcal{T}^{t+1}.$$

Therefore, the truncated core tensor can be updated with the help of the auxiliary tensor $\mathcal{T}^t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times R_N}$, which can save both space cost and time cost.

The whole procedure of iT-HOSVD is presented in Algorithm 2.

Property 2 Assume the time complexity of computing the SVD of an $m \times n$ matrix is $O(mn \min(m, n))$. Then the time complexity of iT-HOSVD is

$$O\left(R_N^2 t + R_N(R_1 + R_N)J + \sum_{n=1}^{N-1} \left(I_n J + R_n I_n^2 \right) \right).$$

Proof First, we consider the cost of updating the $n(1 \le n \le N-1)$th mode. In line 2 of Algorithm 2, the cost of obtaining $\mathbf{X}_n^t$ is $O(R_n J)$. The cost of line 3 for computing $\mathbf{Y}_n^t$ is $O(R_n J)$. In line 4, the computation of the QR decomposition of $\mathbf{Y}_n^t$ is $O(I_n J)$. The cost of obtaining $\mathbf{D}_n^t$ in line 5 is $O(I_n J)$. The cost of computing the SVD of $\mathbf{D}_n^t$ in line 6 is $O(I_n J + R_n I_n^2)$. The cost of line 8 for computing $\bar{\mathbf{U}}_n^{t+1}$ is $O(R_n I_n^2)$. So the

Algorithm 2: incremental truncated HOSVD (iT-HOSVD)

Input: $\bar{\mathbf{U}}_n^t$ and $\bar{\mathbf{\Sigma}}_n^t$ for $n = 1, \ldots, N$; $\bar{\mathcal{V}}_N^t$; $\mathcal{T}^t$; $\mathcal{B}^t$

Output: $\bar{\mathbf{U}}_n^{t+1}$ and $\bar{\mathbf{\Sigma}}_n^{t+1}$ for $n = 1, \ldots, N$; $\bar{\mathcal{V}}_N^{t+1}$; $\mathcal{T}^{t+1}$; $\bar{\mathcal{S}}^{t+1}$

1 **for** $n = 1, \ldots, N-1$ **do**

2 $\quad \mathbf{X}_n^t \leftarrow \bar{\mathbf{U}}_n^{t\top} \mathbf{B}_{(n)}^t$

3 $\quad \mathbf{Y}_n^t \leftarrow \mathbf{B}_{(n)}^t - \bar{\mathbf{U}}_n^t \mathbf{X}_n^t$

4 $\quad \begin{bmatrix} \mathbf{P}_n^t & \sim \end{bmatrix} = \mathtt{qr}\,(\mathbf{Y}_n^t)$ ▷ Compute the QR decomposition of $\mathbf{Y}_n^t$

5 $\quad \mathbf{D}_n^t \leftarrow \begin{bmatrix} \bar{\mathbf{\Sigma}}_n^t & \mathbf{X}_n^t \\ \mathbf{0} & \mathbf{P}_n^{t\top} \mathbf{Y}_n^t \end{bmatrix}$

6 $\quad [\mathbf{G}_n^t, \mathbf{\Lambda}_n^t, \mathbf{H}_n^t] = \mathtt{SVD}(\mathbf{D}_n^t)$ ▷ Compute the SVD of $\mathbf{D}_n^t \in \mathbb{R}^{I_n \times (\frac{J}{I_n} + R_n)}$

7 $\quad \mathbf{\Phi}_n^t \leftarrow \mathbf{G}_n^t(1 : R_n, 1 : R_n)$, $\mathbf{\Psi}_n^t \leftarrow \mathbf{G}_n^t(R_n + 1 : I_n, 1 : R_n)$

8 $\quad \bar{\mathbf{U}}_n^{t+1} \leftarrow \bar{\mathbf{U}}_n^t \mathbf{\Phi}_n^t + \mathbf{P}_n^t \mathbf{\Psi}_n^t$

9 $\quad \bar{\mathbf{\Sigma}}_n^{t+1} \leftarrow \mathbf{\Lambda}_n^t(1 : R_n, 1 : R_n)$

10 **end**

11 $[\mathbf{G}_N^t, \mathbf{\Lambda}_N^t, \mathbf{H}_N^t] = \mathtt{iSVD}(\bar{\mathbf{\Sigma}}_N^t, \bar{\mathbf{V}}_N^t, \mathrm{vec}(\mathcal{B}^t))$

12 $\mathbf{\Phi}_N^t \leftarrow \mathbf{G}_N^t(1 : R_N, 1 : R_N)$, $\mathbf{\Psi}_N^t \leftarrow \mathbf{G}_N^t(R_N + 1, 1 : R_N)$

13 $\bar{\mathbf{U}}_N^{t+1} \leftarrow \begin{bmatrix} \bar{\mathbf{U}}_N^t \mathbf{\Phi}_N^t \\ \mathbf{\Psi}_N^t \end{bmatrix}$

14 $\bar{\mathbf{\Sigma}}_N^{t+1} \leftarrow \mathbf{\Lambda}_N^t(1 : R_N, 1 : R_N)$

15 $\bar{\mathbf{V}}_N^{t+1} \leftarrow \begin{bmatrix} \bar{\mathbf{V}}_N^t & \mathbf{p} \end{bmatrix} \cdot \mathbf{H}_N^t(:, 1 : R_N)$

16 $\mathcal{T}^{t+1} \leftarrow \mathbf{\Phi}_N^{t\top} \cdot_N \mathcal{T}^t + \mathbf{\Psi}_N^{t\top} \cdot_N \mathcal{B}^t$

17 $\bar{\mathcal{S}}^{t+1} \leftarrow \left(\bar{\mathbf{U}}_1^{t+1\top}, \ldots, \bar{\mathbf{U}}_{N-1}^{t+1\top} \right) \cdot \mathcal{T}^{t+1}$

entire cost for updating the nth mode is $O(I_n J + R_n I_n^2)$ and overall cost for updating the first $N-1$ modes is $O\left(\sum_{n=1}^{N-1} I_n J + R_n I_n^2\right)$.

Second, we consider the cost of updating the Nth mode. The cost of computing the SVD of $\mathbf{D}_N^t$ in line 11 is $O(R_N J)$ (see Sect. 2.2). The cost of line 13 for computing $\bar{\mathbf{U}}_N^{t+1}$ is $O(R_N^2(I_N + t))$. The cost of line 15 for computing $\bar{\mathbf{V}}_N^{t+1}$ is $O(R_N^2 J)$. So the entire cost for updating the Nth mode is $O(R_N^2 J + R_N^2(I_N + t))$.

At last, the total cost of lines 16 and 17 is $O(R_N^2 J + R_1 R_N J)$.

Combining the three parts above yields the final time cost. □

Remark 2 As mentioned in the Introduction, in previous works on the truncated HOSVD, the factor matrices are updated by incremental methods, while the core tensor is updated via a batch-mode computation. For example, i-HOSVD proposed in [30] updates the factor matrices with the same methods as Algorithm 2, but updates the core tensor simply by matrix-tensor products, i.e., lines 16 and 17 of Algorithm 2 are changed into

$$\bar{\mathcal{S}}^{t+1} \leftarrow \left(\bar{\mathbf{U}}_1^{t+1\top}, \cdots, \bar{\mathbf{U}}_N^{t+1\top} \right) \cdot \mathcal{A}^{t+1}.$$

Corresponding to Property 2, the time complexity of this algorithm is

$$O\left(R_N(R_N+J)t + R_N(R_1+R_N+I_N)J + \sum_{n=1}^{N-1}\left(I_nJ + R_nI_n^2\right)\right).$$

3.2 iST-HOSVD

For the ST-HOSVD, $\bar{\mathbf{U}}_{\pi_n}^{t+1}$ has relations with $\bar{\mathbf{U}}_{\pi_1}^{t+1}, \ldots, \bar{\mathbf{U}}_{\pi_{n-1}}^{t+1}$, where $\boldsymbol{\pi}$ is the processing ordering. To design an incremental algorithm to update all factor matrices of the ST-HOSVD is difficult. By the ST-HOSVD procedure (3), we know that the dominant step in time cost is the computation of $\bar{\mathbf{U}}_{\pi_1}^{t+1}$. Updating the first factor matrix (in the processing ordering) can still save a lot cost.

Suppose we choose the τth mode in which the factor matrix is updated in the incremental style. After obtaining $\bar{\mathbf{U}}_{\tau}^{t+1}$, we need to compute the rank-$(R_1, \ldots, R_N)$ approximation of $\mathcal{T}^{t+1} := \bar{\mathbf{U}}_{\tau}^{t+1^\top} \cdot_\tau \mathcal{A}^{t+1}$ by the ST-HOSVD. Two crucial questions are how to choose τ and how to update the τth mode. Here we choose updating the Nth mode by the SVD updating for the following two reasons:

1. Using the SVD updating to update the Nth mode is with a very cheap cost; see Property 2.
2. $\mathcal{T}^{t+1}$ would have a fixed size of $I_1 \times \cdots \times I_{N-1} \times R_N$ and can be updated with the results from the last step.

Updating the Nth mode is in the framework of Sect. 2.2. The tensor $\mathcal{T}^{t+1}$ can be updated by (15). We call this method iST-HOSVD for short and the whole procedure is presented in Algorithm 3.

Following Property 1 and Property 2, we have the following property.

Property 3 Assume the time complexity of computing the SVD of an $m \times n$ matrix is $O(mn\min(m,n))$. The processing ordering of $\mathcal{T}^{t+1}$ for the first $N-1$ modes is $[1, \ldots, N-1]$. Then the time complexity of iST-HOSVD is

$$O\left(R_N^2 t + R_N(R_1I_2/I_1 + R_N + I_1)J\right).$$

3.3 Cost comparison

We compare the computational costs of the proposed algorithms against T-HOSVD, ST-HOSVD, R-HOSVD [33], and i-HOSVD [30]. R-HOSVD is a randomized algorithm, and as mentioned in the Introduction, it is a batch-mode algorithm. i-HOSVD has been introduced in Remark 2.

In terms of space cost, T-HOSVD, ST-HOSVD, R-HOSVD and i-HOSVD need to store the old dataset, the new dataset, and outputs, while iT-HOSVD and iST-HOSVD only need to store the new dataset and outputs. The main time cost of R-HOSVD is for the Tucker sketch ([33, Algorithm 4.1]). Suppose the sketch size parameter is

Algorithm 3: incremental ST-HOSVD based on SVD updating (iST-HOSVD)

Input: $\bar{\mathbf{U}}_N^t, \bar{\mathbf{\Sigma}}_N^t, \bar{\mathbf{V}}_N^t; \mathcal{T}^t; \mathcal{B}^t$

Output: $\bar{\mathbf{U}}_n^{t+1}$ for $n = 1, \ldots, N$; $\bar{\mathbf{\Sigma}}_N^{t+1}, \bar{\mathbf{V}}_N^{t+1}; \mathcal{T}^{t+1}; \bar{\mathcal{S}}^{t+1}$

$[\mathbf{G}_N^t, \mathbf{\Lambda}_N^t, \mathbf{H}_N^t] = \texttt{iSVD}(\bar{\mathbf{\Sigma}}_N^t, \bar{\mathbf{V}}_N^t, \text{vec}(\mathcal{B}^t))$

$\mathbf{\Phi}_N^t \leftarrow \mathbf{G}_N^t(1 : R_N, 1 : R_N), \mathbf{\Psi}_N^t \leftarrow \mathbf{G}_N^t(R_N + 1, 1 : R_N)$

$\bar{\mathbf{U}}_N^{t+1} \leftarrow \begin{bmatrix} \bar{\mathbf{U}}_N^t \mathbf{\Phi}_N^t \\ \mathbf{\Psi}_N^t \end{bmatrix}$

$\bar{\mathbf{\Sigma}}_N^{t+1} \leftarrow \mathbf{\Lambda}_N^t(1 : R_N, 1 : R_N)$

$\bar{\mathbf{V}}_N^{t+1} \leftarrow \begin{bmatrix} \bar{\mathbf{V}}_N^t & \mathbf{p} \end{bmatrix} \cdot \mathbf{H}_N^t(:, 1 : R_N)$

$\mathcal{T}^{t+1} \leftarrow \mathbf{\Phi}_N^{t\top} \cdot_N \mathcal{T}^t + \mathbf{\Psi}_N^{t\top} \cdot_N \mathcal{B}^t$

$\boldsymbol{\pi} \leftarrow$ processing ordering for $[1, \ldots, N-1]$

$\mathcal{T} \leftarrow \mathcal{T}^{t+1}$

for $n = 1, \ldots, N-1$ **do**

  $\mathbf{T}_{(\pi_n)} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_1 & \\ & \mathbf{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^\top \\ \mathbf{V}_2^\top \end{bmatrix}$, with $\mathbf{U}_1 \in \mathbb{R}^{I_{\pi_n} \times R_{\pi_n}}$

  ▷ Compute the SVD of $\mathbf{T}_{(\pi_n)}$

  $\bar{\mathbf{U}}_{\pi_n}^{t+1} \leftarrow \mathbf{U}_1$

  $\mathbf{T}_{(\pi_n)} \leftarrow \mathbf{\Sigma}_1 \mathbf{V}_1^\top$

end

$\bar{\mathcal{S}}^{t+1} \leftarrow \mathcal{T}$

$(K_1, \ldots, K_N)$. Then the cost of the Tucker sketch is

$$O\left(\sum_{n=1}^{N} K_n J(I_N + t)\right).$$

The requirement of the sketch size parameter is $K_n \geq R_n, n = 1, \ldots, N$. The time costs of the other four algorithms have been discussed before. We summarize the costs in Table 1. For simplicity, we suppose

$$I_n = I, \quad R_n = R, \quad K_n = K, \quad n = 1, \ldots, N.$$

The time costs of all algorithms would increase as t increases, and all time costs increase linearly when t is sufficiently big. However, the increasing speeds differ greatly for different algorithms. The terms related to t of the time costs of iT-HOSVD and iST-HOSVD are the same: $R^2 t$, which is much smaller than the other algorithms.

4 Approximation error analysis

We analyze the approximation errors for the two incremental algorithms, and give some discussions on the obtained results. For convenience, we introduce some notations. Let $\mathbf{A}_{(n)}^t$ be the mode-n unfolding matrix of $\mathcal{A}^t$. We let

$$\mathbf{A}_{(n),R_n}^t \text{ be the best rank-}R_n \text{ approximation of } \mathbf{A}_{(n)}^t, \quad n = 1, \ldots, N. \tag{16}$$

Table 1 Costs comparison

Algorithm	Space	Time
T-HOSVD	$(I^{N-1}+R)t+I^N+R^N+NRI$	$O\Big(\big((N-1)I+\min(I+t,I^{N-1})\big)I^{N-1}(I+t)\Big)$
ST-HOSVD	$(I^{N-1}+R)t+I^N+R^N+NRI$	$O\Big((I+2R)I^{N-1}(I+t)\Big)$
R-HOSVD	$(I^{N-1}+R)t+I^N+R^N+NRI$	$O\Big(NKI^{N-1}(I+t)\Big)$
i-HOSVD	$(I^{N-1}+R)t+I^N+R^N+NRI$	$O\Big(R(R+I^{N-1})t+R(2R+I)I^{N-1}+(N-1)(I^N+RI^2)\Big)$
iT-HOSVD	$Rt+(2R+1)I^{N-1}+R^N+NRI+(N-1)R$	$O\Big(R^2t+2R^2I^{N-1}+(N-1)(I^N+RI^2)\Big)$
iST-HOSVD	$Rt+(2R+1)I^{N-1}+R^N+NRI+(N-1)R$	$O\Big(R^2t+R(2R+I)I^{N-1}\Big)$

The initialization data $\mathcal{A}^0$ is an Nth-order tensor of size $I\times\cdots\times I$, the truncation rank is $(R,\ldots,R)$, and the sketch size parameter of R-HOSVD is $(K,\ldots,K)$

In Algorithm 2 and 3, the truncated SVD of $\begin{bmatrix} \bar{\mathbf{U}}_N^t \bar{\mathbf{\Sigma}}_N^t \bar{\mathbf{V}}_N^{t\top} \\ \mathrm{vec}(\mathcal{B}^t)^\top \end{bmatrix}$ of rank R_N is

$$\begin{bmatrix} \bar{\mathbf{U}}_N^t \bar{\mathbf{\Sigma}}_N^t \bar{\mathbf{V}}_N^{t\top} \\ \mathrm{vec}(\mathcal{B}^t)^\top \end{bmatrix} \approx \bar{\mathbf{U}}_N^{t+1} \bar{\mathbf{\Sigma}}_N^{t+1} \bar{\mathbf{V}}_N^{t+1\top}.$$

In Algorithm 2, we do not need to compute $\bar{\mathbf{V}}_n^{t+1}$ for $n = 1, \ldots, N-1$. Actually, $\bar{\mathbf{V}}_n^{t+1}$ can be obtained by (12) easily. By (9), (12) and (13), we assume that the truncated SVD of $\begin{bmatrix} \bar{\mathbf{U}}_n^t \bar{\mathbf{\Sigma}}_n^t \bar{\mathbf{V}}_n^{t\top} & \mathbf{B}_{(n)}^t \end{bmatrix}$ of rank R_n is

$$\begin{bmatrix} \bar{\mathbf{U}}_n^t \bar{\mathbf{\Sigma}}_n^t \bar{\mathbf{V}}_n^{t\top} & \mathbf{B}_{(n)}^t \end{bmatrix} \approx \bar{\mathbf{U}}_n^{t+1} \bar{\mathbf{\Sigma}}_n^{t+1} \bar{\mathbf{W}}_n^{t+1\top}. \tag{17}$$

With the same column permutation from $\begin{bmatrix} \mathbf{A}_{(n)}^t & \mathbf{B}_{(n)}^t \end{bmatrix}$ to $\mathbf{A}_{(n)}^{t+1}$, we can get $\bar{\mathbf{V}}_n^{t+1\top}$ from $\bar{\mathbf{W}}_n^{t+1\top}$. We define

$$\bar{\mathbf{A}}_{(n)}^t := \bar{\mathbf{U}}_n^t \bar{\mathbf{\Sigma}}_n^t \bar{\mathbf{V}}_n^{t\top}, n = 1, \ldots, N.$$

By definition,

$$\bar{\mathbf{A}}_{(n)}^0 = \mathbf{A}_{(n),R_n}^0, \quad n = 1, \ldots, N. \tag{18}$$

Before we discuss the approximation errors of the two incremental algorithms, we review the existing results on approximation errors of the truncated HOSVD. R-HOSVD is a randomized algorithm, and its approximation error is discussed in [33]. Its approximation error depends on the distribution of the singular values of the unfolding matrices. In practical applications, the approximation error of R-HOSVD is greater than those of T-HOSVD and ST-HOSVD. The error bounds of T-HOSVD and ST-HOSVD have been given in [37, Corollary 5.2 and Theorem 6.5]. Let $\bar{\mathcal{A}}^t$ and $\hat{\mathcal{A}}^t$ be the rank-$(R_1, \ldots, R_N)$ T-HOSVD and ST-HOSVD of $\mathcal{A}$, respectively. Then,

$$\begin{aligned} \|\mathcal{A}^t - \bar{\mathcal{A}}^t\| &\le \sqrt{\sum_{n=1}^N \left\| \mathbf{A}_{(n)}^t - \mathbf{A}_{(n),R_n}^t \right\|^2}, \\ \|\mathcal{A}^t - \hat{\mathcal{A}}^t\| &\le \sqrt{\sum_{n=1}^N \left\| \mathbf{A}_{(n)}^t - \mathbf{A}_{(n),R_n}^t \right\|^2}. \end{aligned} \tag{19}$$

The following two theorems guarantee the performance of the proposed incremental algorithms.

Theorem 2 *Suppose the result of iT-HOSVD for $\mathcal{A}^t$ is $\left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t$. We have*

$$\begin{aligned}\left\|\mathcal{A}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t\right\| \leq & \sum_{n=1}^{N} \left\|\mathbf{A}_{(n)}^0 - \mathbf{A}_{(n),R_n}^0\right\| \\ & + \sum_{k=0}^{t-1}\sum_{n=1}^{N-1} \left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^k]\mathcal{B}^k\right\| + \sum_{k=0}^{t-1} \left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|,\end{aligned} \tag{20}$$

where $\mathbf{A}_{(n),R_n}^0$ is defined in (16).

Theorem 3 *Suppose the result of iST-HOSVD for $\mathcal{A}^t$ is $\left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t$. We have*

$$\begin{aligned}\left\|\mathcal{A}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t\right\| \leq & \left\|\mathbf{A}_{(N)}^0 - \mathbf{A}_{(N),R_N}^0\right\| + \sum_{k=0}^{t-1} \left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\| \\ & + \sqrt{\sum_{n=1}^{N-1} \left\|\mathbf{A}_{(n)}^t - \mathbf{A}_{(n),R_n}^t\right\|^2},\end{aligned} \tag{21}$$

where $\mathbf{A}_{(n),R_n}^t$ is defined in (16).

Compared to the error bounds (19), the error bounds of the two incremental algorithms are related to the approximation error of previous steps: the error bound (20) of iT-HOSVD is related to

$$\left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^k]\mathcal{B}^k\right\| \text{ and } \left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|, \quad n = 1, \ldots, N-1, k = 0, 1, \ldots, t-1,$$

and the error bound (21) of iST-HOSVD is related to

$$\left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|, \quad k = 0, 1, \ldots, t-1.$$

By noting (8), $\left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^k]\mathcal{B}^k\right\|$ is the approximation error of $\mathbf{B}_{(n)}^k$ by the previous basis $\bar{\mathbf{U}}_n^k$, and $\left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|$ is the approximation error of $\mathrm{vec}(\mathcal{B}^k)$ by the previous basis $\bar{\mathbf{V}}_N^k$. This makes sense for the computation in an incremental style. If there is a great variation between the new slice $\mathcal{B}^k$ and the old data, this approximation error would be great. The situation is similar for $\left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|$ in (21). Hence, the approximation errors of iT-HOSVD and iST-HOSVD depend heavily on the character of the variation of the data.

To prove Theorems 2 and 3, we first give a result related to unfolding matrices. In fact, this result is the approximation error of the matrix incremental SVD of appending several columns.

Lemma 1 *For Algorithm 2, we have*

$$\begin{aligned}\left\|\mathbf{A}_{(n)}^{t+1}-\bar{\mathbf{A}}_{(n)}^{t+1}\right\| &\le \left\|\mathbf{A}_{(n)}^{t}-\bar{\mathbf{A}}_{(n)}^{t}\right\| + \left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^t]\mathcal{B}^t\right\| \quad \forall n=1,\ldots,N-1,\\ \left\|\mathbf{A}_{(N)}^{t+1}-\bar{\mathbf{A}}_{(N)}^{t+1}\right\| &\le \left\|\mathbf{A}_{(N)}^{t}-\bar{\mathbf{A}}_{(N)}^{t}\right\| + \left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^t]\mathrm{vec}(\mathcal{B}^t)\right\|.\end{aligned}$$

Proof For $n=1,\ldots,N-1$, combining (9), (12) and (17) yields that

$$\begin{aligned}\left\|\mathbf{A}_{(n)}^{t+1}-\bar{\mathbf{A}}_{(n)}^{t+1}\right\| &= \left\|\begin{bmatrix}\mathbf{A}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \bar{\mathbf{U}}_n^{t+1}\bar{\mathbf{\Sigma}}_n^{t+1}\bar{\mathbf{W}}_n^{t+1^\top}\right\|\\ &\le \left\|\begin{bmatrix}\mathbf{A}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \begin{bmatrix}\bar{\mathbf{A}}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix}\right\| + \left\|\begin{bmatrix}\bar{\mathbf{A}}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \bar{\mathbf{U}}_n^{t+1}\bar{\mathbf{\Sigma}}_n^{t+1}\bar{\mathbf{W}}_n^{t+1^\top}\right\|.\end{aligned}$$

We have

$$\left\|\begin{bmatrix}\mathbf{A}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \begin{bmatrix}\bar{\mathbf{A}}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix}\right\| = \left\|\mathbf{A}_{(n)}^{t}-\bar{\mathbf{A}}_{(n)}^{t}\right\|,$$

and $\left\|\begin{bmatrix}\bar{\mathbf{A}}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \bar{\mathbf{U}}_n^{t+1}\bar{\mathbf{\Sigma}}_n^{t+1}\bar{\mathbf{W}}_n^{t+1^\top}\right\|$ is just the approximation error of the best rank-R_n approximation of $\mathbf{D}_n^t$ defined in (11). Since $\begin{bmatrix}\bar{\mathbf{\Sigma}}_n^t & \mathbf{X}_n^t\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$ is rank-R_n, the approximation error of the best rank-R_n approximation of $\mathbf{D}_n^t$ is less than $\|\mathbf{P}_n^{t^\top}\mathbf{Y}_n^t\|$. On the other hand, for (10), since $\mathbf{P}_n^t$ is an orthonormal basis of the column space of $\mathbf{Y}_n^t$, we have

$$\left\|\mathbf{P}_n^{t^\top}\mathbf{Y}_n^t\right\| = \|\mathbf{Y}_n^t\| = \left\|\mathbf{B}_{(n)}^t - \bar{\mathbf{U}}_n^t\bar{\mathbf{U}}_n^{t^\top}\mathbf{B}_{(n)}^t\right\| = \left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^t]\mathcal{B}^t\right\| \quad \forall n=1,\ldots,N-1.$$

Hence,

$$\left\|\begin{bmatrix}\bar{\mathbf{A}}_{(n)}^{t} & \mathbf{B}_{(n)}^{t}\end{bmatrix} - \bar{\mathbf{U}}_n^{t+1}\bar{\mathbf{\Sigma}}_n^{t+1}\bar{\mathbf{W}}_n^{t+1^\top}\right\| \le \left\|\mathbf{P}_n^{t^\top}\mathbf{Y}_n^t\right\| = \left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^t]\mathcal{B}^t\right\|.$$

This completes the proof.

The case $n=N$ can be proved similarly. □

The following result is an immediate consequence of the preceding lemma, where (18) has been used.

Corollary 1 *For Algorithm 2, we have*

$$\begin{aligned}\left\|\mathbf{A}_{(n)}^{t+1}-\bar{\mathbf{A}}_{(n)}^{t+1}\right\| &\le \left\|\mathbf{A}_{(n)}^{0}-\mathbf{A}_{(n),R_n}^{0}\right\| + \sum_{k=0}^{t}\left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^k]\mathcal{B}^k\right\| \quad \forall n=1,\ldots,N-1,\\ \left\|\mathbf{A}_{(N)}^{t+1}-\bar{\mathbf{A}}_{(N)}^{t+1}\right\| &\le \left\|\mathbf{A}_{(N)}^{0}-\mathbf{A}_{(N),R_N}^{0}\right\| + \sum_{k=0}^{t}\left\|\mathscr{P}_1^{\perp}[\bar{\mathbf{V}}_N^k]\mathrm{vec}(\mathcal{B}^k)\right\|.\end{aligned} \tag{22}$$

Now we can prove Theorems 2 and 3.

Proof of Theorem 2 It follows from (7) that

$$\left\|\mathcal{A}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t\right\| \leq \sum_{n=1}^{N} \left\|\mathscr{P}_n^{\perp}[\bar{\mathbf{U}}_n^t]\mathcal{A}^t\right\| \overset{(6)}{=} \sum_{n=1}^{N} \left\|\mathbf{A}_{(n)}^t - \bar{\mathbf{A}}_{(n)}^t\right\|.$$

Combining (22) with the above equation yields the result. □

Proof of Theorem 3 First, by the algorithm and (7), we have

$$\left\|\mathcal{A}^t - \bar{\mathbf{U}}_N^t \cdot_N \mathcal{T}^t\right\| \leq \left\|\mathscr{P}_N^{\perp}[\bar{\mathbf{U}}_N^t]\mathcal{A}^t\right\| \overset{(6)}{=} \left\|\mathbf{A}_{(N)}^t - \bar{\mathbf{A}}_{(N)}^t\right\|.$$

Second, by the algorithm, $\left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_{N-1}^t\right) \cdot \bar{\mathcal{S}}^t$ is the rank-$(R_1, \ldots, R_N)$ approximation of $\mathcal{T}^t$ by the ST-HOSVD. By [37, Theorem 6.5], we have

$$\begin{aligned}&\left\|\bar{\mathbf{U}}_N^t \cdot_N \left(\mathcal{T}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_{N-1}^t\right) \cdot \bar{\mathcal{S}}^t\right)\right\| \overset{(4)}{=} \left\|\mathcal{T}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_{N-1}^t\right) \cdot \bar{\mathcal{S}}^t\right\| \\ &\leq \sqrt{\sum_{n=1}^{N-1} \left\|\mathbf{A}_{(n)}^t - \mathbf{A}_{(n),R_n}^t\right\|^2}.\end{aligned}$$

Combining the above two equations yields that

$$\begin{aligned}&\left\|\mathcal{A}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_N^t\right) \cdot \bar{\mathcal{S}}^t\right\| \\ &\leq \left\|\mathcal{A}^t - \bar{\mathbf{U}}_N^t \cdot_N \mathcal{T}^t\right\| + \left\|\bar{\mathbf{U}}_N^t \cdot_N \left(\mathcal{T}^t - \left(\bar{\mathbf{U}}_1^t, \ldots, \bar{\mathbf{U}}_{N-1}^t\right) \cdot \bar{\mathcal{S}}^t\right)\right\| \\ &\leq \left\|\mathbf{A}_{(N)}^t - \bar{\mathbf{A}}_{(N)}^t\right\| + \sqrt{\sum_{n=1}^{N-1} \left\|\mathbf{A}_{(n)}^t - \mathbf{A}_{(n),R_n}^t\right\|^2}. \qquad (23)\end{aligned}$$

The update of the Nth mode of iST-HOSVD is the same as that of iT-HOSVD. Combining (22) and (23) yields the result. □

5 Numerical experiments

In this section, we compare the proposed algorithms with three batch-mode algorithms T-HOSVD, ST-HOSVD and R-HOSVD [33], and one incremental algorithm i-HOSVD [30]. We use the relative error (RErr) to evaluate results. For the truncated HOSVD $\bar{\mathcal{A}} = \left(\bar{\mathbf{U}}_1, \ldots, \bar{\mathbf{U}}_N\right) \cdot \bar{\mathcal{S}}$ of $\mathcal{A}$, the RErr is defined as

$$\text{RErr} = \frac{\|\mathcal{A} - \bar{\mathcal{A}}\|}{\|\mathcal{A}\|}.$$

All experiments are performed on MATLAB R2023a with tensor Toolbox, version 3.0 [1] on a work station (Intel Core i7-10875H @2.3Hz, 32 G RAM).

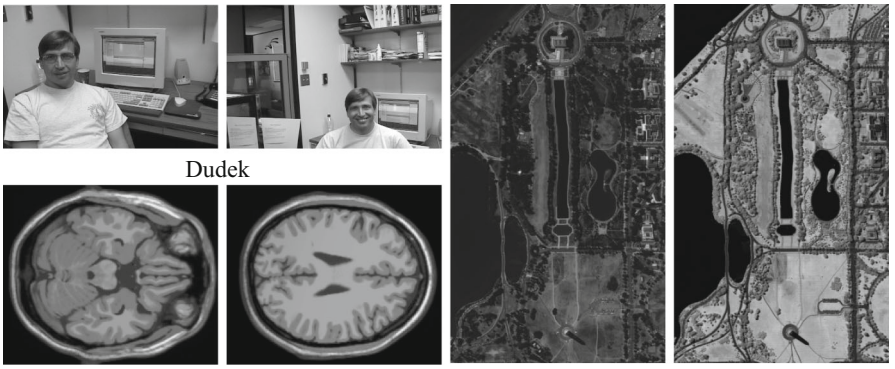

Fig. 1 Two slices of each third-order tensor

Suppose the truncation rank is $(R_1, \ldots, R_N)$. For R-HOSVD, we use the suggested sketch size parameter used in [33] for each test:

$$K_n = 2R_n + 1, \quad n = 1, \ldots, N.$$

5.1 Results on real-world datasets

As mentioned in Sect. 4, the approximation errors of iT-HOSVD and iST-HOSVD depend heavily on the character of the variation of the data. We will test tensors with different variation features. For third-order tensors, we test a video "Dudek", a hyperspectral image "WDC" and an MRI "Brain".[3] For fourth-order tensors, we test three videos "Highway", "Corridor", and "Bridge".[4] In the direction of the last mode, "Dudek" and "Highway" are rapidly changing, "MRI" and "Corridor" are moderately changing, and "WDC" and "Bridge" are relatively stable. We show two slices of each tensor in Figs. 1 and 2. The detailed information of each test tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times L}$ is given in Table 2.

We choose the first 10% slices, i.e., $\mathcal{A}^0 = \mathcal{A}\left(:, \ldots, :, 1 : \left\lfloor \frac{L}{10} \right\rfloor\right)$, as the initialization data. We compute the full HOSVD of $\mathcal{A}^0$ and use it to obtain the initializations of all incremental algorithms. After initialization and setting, the remaining 90% data are appended to the existing tensor one slice at a time. There are $T := L - \left\lfloor \frac{L}{10} \right\rfloor$ slices that will be appended. At the t-th step, we compute the truncated HOSVD of

$$\mathcal{A}^t = \mathcal{A}\left(:, \ldots, :, 1 : \left\lfloor \frac{L}{10} \right\rfloor + t\right), \quad t = 1, 2, \ldots, T$$

[3] "Dudek" is widely used for testing visual tracking algorithm [15, 27] and available on http://www.cs.toronto.edu/~dross/ivt/. "WDC" is short for *Washington DC Mall* and available on https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html. "Brain" is from BrainWeb [6] and available at http://brainweb.bic.mni.mcgill.ca/brainweb/selection_normal.html.

[4] "Highway" and "Bridge" are from the video trace library [28] and available on http://trace.eas.asu.edu/yuv/. "Corridor" is used in [15] and available on https://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.

Fig. 2 Two slices of each fourth-order tensor

Table 2 Details of test tensors

Tensor	$I_1 \times \cdots \times I_{N-1} \times L$	$J = \prod_{n=1}^{N-1} I_n$	J/L
Dudek	$480 \times 720 \times 573$	345,600	603.1
Brain	$181 \times 217 \times 181$	39,277	181.0
WDC	$500 \times 300 \times 191$	150,000	785.3
Highway	$176 \times 144 \times 3 \times 2000$	76,032	38.0
Corridor	$288 \times 384 \times 3 \times 2357$	331,776	140.8
Bridge	$176 \times 144 \times 3 \times 2001$	76,032	38.0

by different methods, where the incremental methods are computed based on the result of the $(t-1)$-st step, and the batch-mode methods are computed from scratch. After processing the t-th step, we record the running time c_t (in s) and compute the RErr e_t. After all data are processed, we compute the mean RErr and the mean running time of one step:

$$\text{mean RErr} = \frac{\sum_{t=1}^{T} e_t}{T}, \quad \text{mean running time} = \frac{\sum_{t=1}^{T} c_t}{T}.$$

The experiments are repeated ten times and we show the average results.

First, we focus on the RErr. In Table 3, we show the mean RErr of one step. R-HOSVD performs the worst among all algorithms. Like [37], we can find that ST-HOSVD performs better than T-HOSVD on all test tensors. The results of i-HOSVD are the same as those of iT-HOSVD. iST-HOSVD outperforms iT-HOSVD on all test tensors. The incremental algorithms show very promising results in accuracy: the ratios between the ST-HOSVD RErr's and the RErr's of the incremental algorithms

Table 3 Mean RErr of one step

Tensor	Truncation rank	T-HOSVD	ST-HOSVD	R-HOSVD	i-HOSVD	iT-HOSVD	iST-HOSVD
Dudek	(5,5,5)	3.14e−01	3.11e−01	3.95e−01	3.25e−01 (0.96)	3.25e−01 (0.96)	3.22e−01 (0.97)
	(10,10,10)	2.64e−01	2.61e−01	3.26e−01	2.73e−01 (0.96)	2.73e−01 (0.96)	2.70e−01 (0.97)
	(20,20,20)	2.14e−01	2.12e−01	2.62e−01	2.23e−01 (0.95)	2.23e−01 (0.95)	2.21e−01 (0.96)
	(30,30,30)	1.85e−01	1.84e−01	2.26e−01	1.92e−01 (0.95)	1.92e−01 (0.95)	1.91e−01 (0.96)
Brain	(5,5,5)	3.77e−01	3.70e−01	4.54e−01	3.89e−01 (0.95)	3.89e−01 (0.95)	3.75e−01 (0.99)
	(10,10,10)	2.79e−01	2.77e−01	3.41e−01	2.88e−01 (0.96)	2.88e−01 (0.96)	2.83e−01 (0.98)
	(20,20,15)	2.05e−01	2.03e−01	2.42e−01	2.12e−01 (0.96)	2.12e−01 (0.96)	2.07e−01 (0.98)
	(30,30,15)	1.69e−01	1.69e−01	1.96e−01	1.78e−01 (0.95)	1.78e−01 (0.95)	1.73e−01 (0.97)
WDC	(5,5,5)	2.69e−01	2.68e−01	3.43e−01	2.71e−01 (0.99)	2.71e−01 (0.99)	2.69e−01 (1.00)
	(10,10,10)	2.39e−01	2.37e−01	2.95e−01	2.44e−01 (0.97)	2.44e−01 (0.97)	2.37e−01 (1.00)
	(20,20,15)	2.01e−01	1.99e−01	2.48e−01	2.06e−01 (0.96)	2.06e−01 (0.96)	1.99e−01 (1.00)
	(30,30,15)	1.76e−01	1.74e−01	2.17e−01	1.81e−01 (0.96)	1.81e−01 (0.96)	1.74e−01 (1.00)
Hihgway	(5,5,3,5)	9.80e−02	9.70e−02	1.26e−01	1.00e−01 (0.97)	1.00e−01 (0.97)	9.96e−02 (0.97)
	(10,10,3,10)	8.20e−02	8.10e−02	1.03e−01	8.61e−02 (0.94)	8.61e−02 (0.94)	8.43e−02 (0.96)
	(20,20,3,20)	6.85e−02	6.75e−02	8.27e−02	7.16e−02 (0.94)	7.16e−02 (0.94)	7.03e−02 (0.96)
	(30,30,3,30)	5.85e−02	5.80e−02	7.05e−02	6.15e−02 (0.94)	6.15e−02 (0.94)	6.03e−02 (0.96)
Corridor	(5,5,3,5)	1.57e−01	1.56e−01	2.00e−01	1.57e−01 (0.99)	1.57e−01 (0.99)	1.57e−01 (1.00)
	(10,10,3,10)	1.23e−01	1.22e−01	1.53e−01	1.24e−01 (0.98)	1.24e−01 (0.98)	1.23e−01 (0.99)
	(20,20,3,20)	9.00e−02	8.94e−02	1.10e−01	9.19e−02 (0.97)	9.19e−02 (0.97)	9.08e−02 (0.98)
	(30,30,3,30)	7.11e−02	7.07e−02	8.71e−02	7.30e−02 (0.97)	7.30e−02 (0.97)	7.24e−02 (0.98)
Bridge	(5,5,3,5)	9.16e−02	9.15e−02	1.24e−01	9.16e−02 (1.00)	9.16e−02 (1.00)	9.15e−02 (1.00)
	(10,10,3,10)	7.41e−02	7.33e−02	9.71e−02	7.42e−02 (0.99)	7.42e−02 (0.99)	7.34e−02 (1.00)
	(20,20,3,20)	5.89e−02	5.84e−02	7.47e−02	5.90e−02 (0.99)	5.90e−02 (0.99)	5.81e−02 (1.00)
	(30,30,3,30)	5.07e−02	5.00e−02	6.22e−02	5.08e−02 (0.98)	5.08e−02 (0.98)	4.99e−02 (1.00)

For the three incremental algorithms, the ratios between the result of ST-HOSVD and theirs are shown in parenthesis

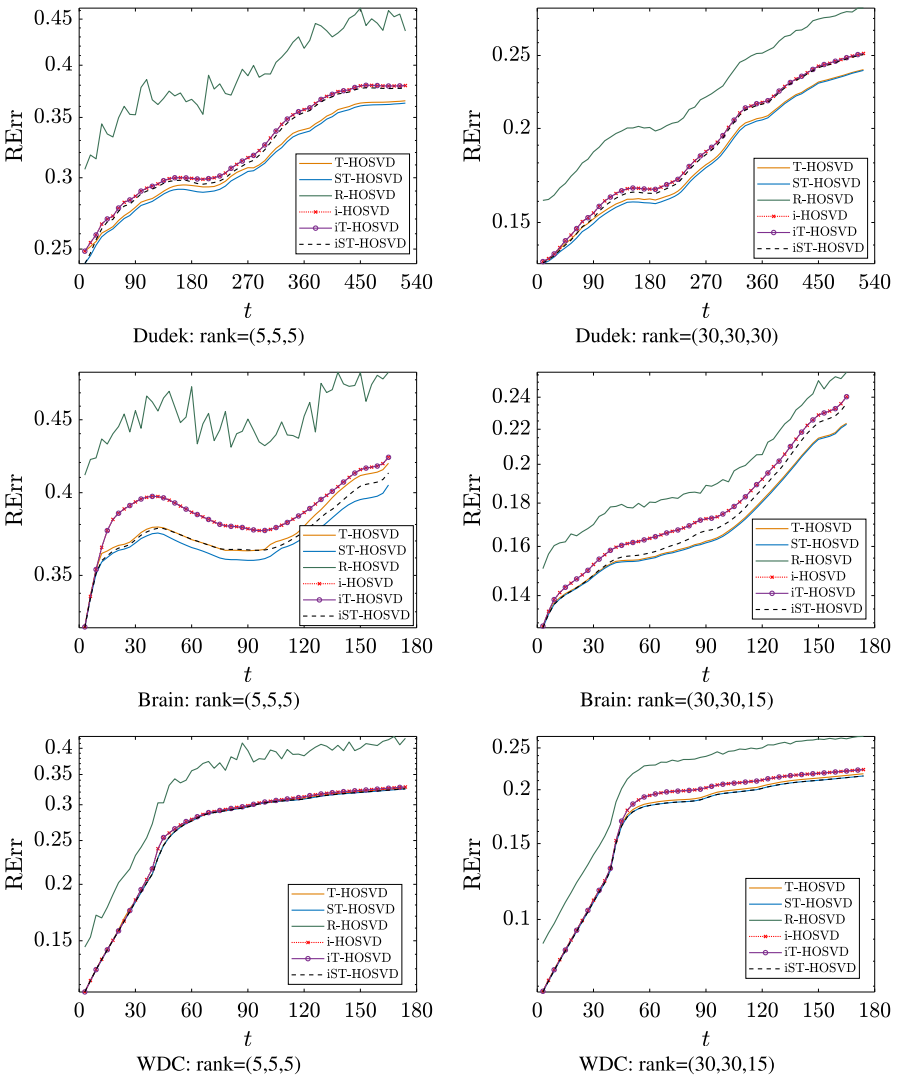

Fig. 3 The comparison results on the relative error for each step: third-order tensors

are greater than 0.94 in all cases, and iST-HOSVD even outperforms the batch-mode algorithms in some cases (see the results for Bridge). The performance of the incremental algorithms is related to the variation features of test tensors: the ratio between the ST-HOSVD RErr and the iT-HOSVD RErr (iST-HOSVD RErr, respectively) is the biggest on the stable tensors "WDC" and "Bridge", while the ratio is relatively small on the rapidly changing tensors "Dudek" and "Highway". Some comparison results on the relative error for each step are shown in Figs. 3 and 4. These results are consistent with the results in Table 3. iST-HOSVD outperforms iT-HOSVD for

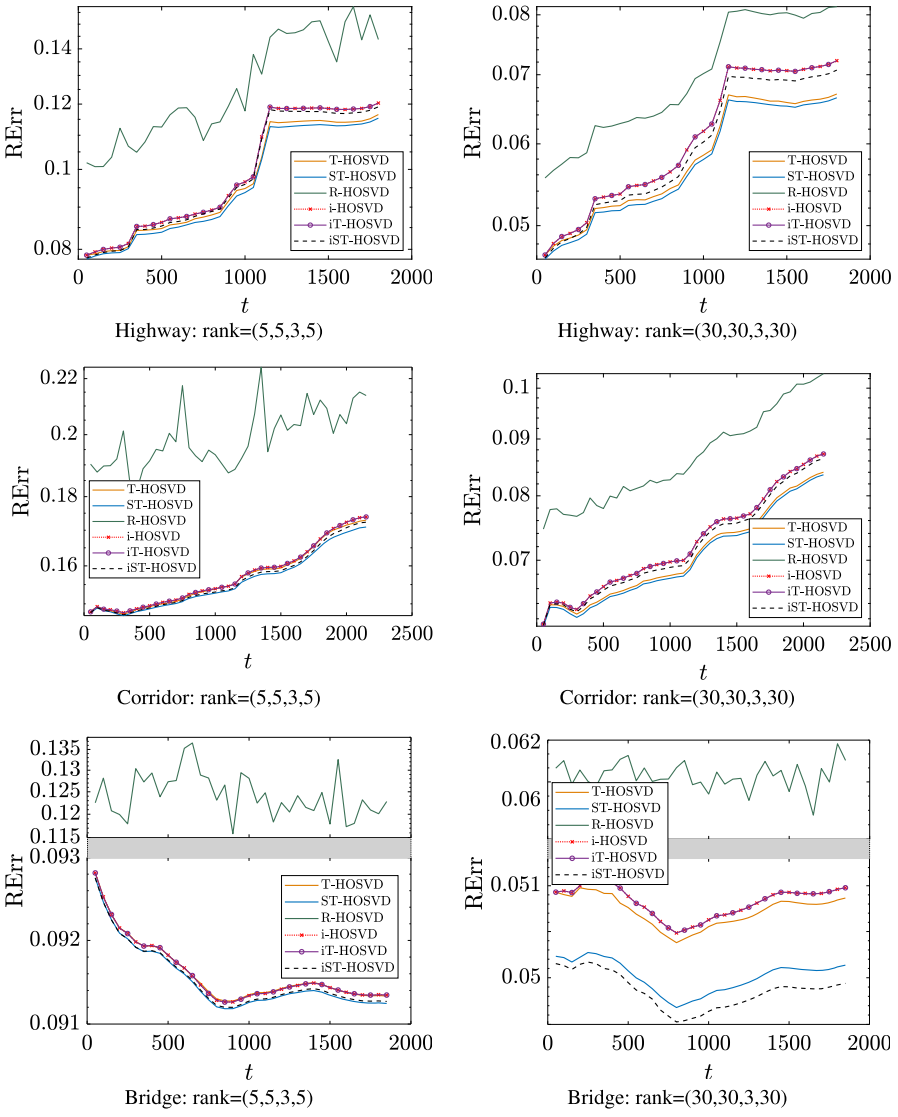

Fig. 4 The comparison results on the relative error for each step: fourth-order tensors

all cases. For the case Bridge with rank=(30,30,3,30), iST-HOSVD even outperforms ST-HOSVD.

Now we focus on the running time. In Table 4, we show the mean running time of one step. The running time of ST-HOSVD is much shorter than that of T-HOSVD, and R-HOSVD is the fastest among the three batch-mode algorithms. We show the ratios between the running time of R-HOSVD and those of the three incremental algorithms. We can find that the incremental algorithms outperform the batch-mode algorithms. As discussed in Sect. 3.3, the time complexities of iT-HOSVD and iST-

Table 4 Mean running time of one step

Tensor	Truncation rank	T-HOSVD	ST-HOSVD	R-HOSVD	i-HOSVD	iT-HOSVD	iST-HOSVD
Dudek	(5,5,5)	1.47e+01	4.61e+00	4.90e−01	3.46e−01 (1.42)	1.24e−01 (3.95)	1.02e−01 (4.80)
	(10,10,10)	1.48e+01	4.70e+00	5.83e−01	3.82e−01 (1.53)	1.48e−01 (3.94)	1.90e−01 (3.07)
	(20,20,20)	1.48e+01	4.81e+00	8.01e−01	4.09e−01 (1.96)	1.94e−01 (4.13)	3.62e−01 (2.21)
	(30,30,30)	1.48e+01	4.94e+00	1.06e+00	4.58e−01 (2.31)	2.44e−01 (4.34)	5.78e−01 (1.83)
Brain	(5,5,5)	2.42e−01	8.64e−02	2.53e−02	2.11e−02 (1.20)	1.67e−02 (1.52)	1.18e−02 (2.14)
	(10,10,10)	2.43e−01	9.08e−02	3.45e−02	2.26e−02 (1.53)	1.96e−02 (1.76)	1.99e−02 (1.73)
	(20,20,15)	2.43e−01	9.54e−02	4.98e−02	2.41e−02 (2.07)	2.23e−02 (2.23)	3.10e−02 (1.61)
	(30,30,15)	2.47e−01	1.03e−01	6.31e−02	2.42e−02 (2.61)	2.36e−02 (2.67)	3.17e−02 (1.99)
WDC	(5,5,5)	1.40e+00	6.11e−01	8.19e−02	6.90e−02 (1.19)	5.34e−02 (1.53)	6.06e−02 (1.35)
	(10,10,10)	1.41e+00	6.19e−01	1.07e−01	7.64e−02 (1.40)	6.50e−02 (1.65)	1.03e−01 (1.04)
	(20,20,15)	1.41e+00	6.24e−01	1.45e−01	8.61e−02 (1.68)	7.83e−02 (1.85)	1.43e−01 (1.01)
	(30,30,15)	1.42e+00	6.42e−01	1.73e−01	8.57e−02 (2.02)	7.78e−02 (2.22)	1.50e−01 (1.15)
Hihgway	(5,5,3,5)	8.18e+00	1.83e+00	4.53e−01	2.06e−01 (2.20)	2.60e−02 (17.42)	1.82e−02 (24.89)
	(10,10,3,10)	8.19e+00	1.96e+00	6.53e−01	2.17e−01 (3.01)	2.91e−02 (22.44)	2.71e−02 (24.10)
	(20,20,3,20)	8.22e+00	2.61e+00	9.73e−01	2.30e−01 (4.23)	3.72e−02 (26.16)	3.91e−02 (24.88)
	(30,30,3,30)	8.20e+00	2.89e+00	1.28e+00	2.50e−01 (5.12)	4.98e−02 (25.70)	6.66e−02 (19.22)
Corridor	(5,5,3,5)	8.70e+01	1.88e+01	2.33e+00	3.31e−01 (7.04)	1.29e−01 (18.06)	7.55e−02 (30.86)
	(10,10,3,10)	8.70e+01	1.89e+01	2.76e+00	4.19e−01 (6.59)	1.67e−01 (16.53)	1.57e−01 (17.58)
	(20,20,3,20)	8.57e+01	1.97e+01	3.69e+00	6.37e−01 (5.79)	2.24e−01 (16.47)	3.19e−01 (11.57)
	(30,30,3,30)	8.77e+01	2.08e+01	7.07e+00	8.42e−01 (8.40)	3.77e−01 (18.75)	5.54e−01 (12.76)
Bridge	(5,5,3,5)	8.12e+00	1.81e+00	4.64e−01	2.12e−01 (2.19)	2.35e−02 (19.74)	1.81e−02 (25.64)
	(10,10,3,10)	7.86e+00	1.82e+00	6.20e−01	2.18e−01 (2.84)	2.53e−02 (24.51)	2.32e−02 (26.72)
	(20,20,3,20)	8.09e+00	2.16e+00	9.98e−01	2.30e−01 (4.34)	3.15e−02 (31.68)	4.47e−02 (22.33)
	(30,30,3,30)	7.98e+00	2.30e+00	1.32e+00	2.50e−01 (5.28)	3.44e−02 (38.37)	6.45e−02 (20.47)

For the three incremental algorithms, the ratios between the result of R-HOSVD and theirs are shown in parenthesis

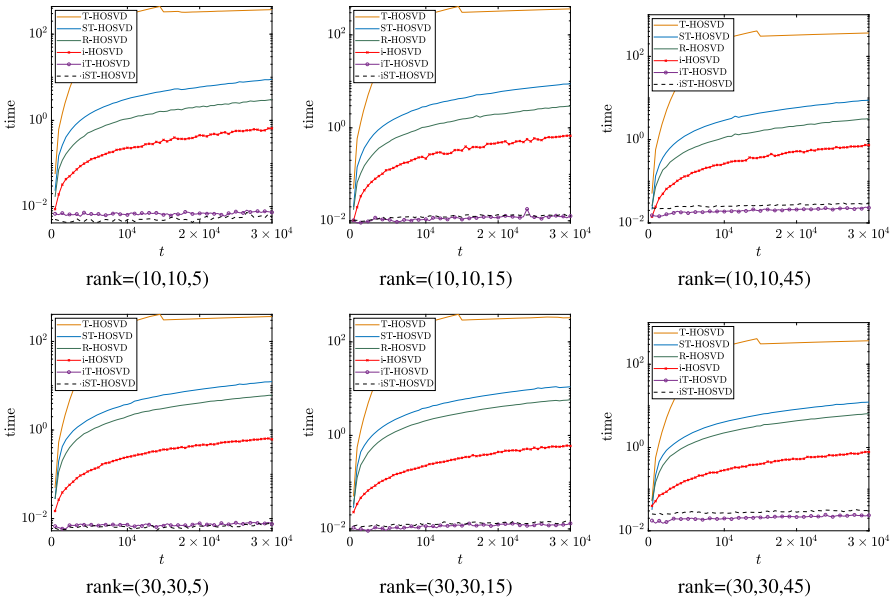

Fig. 5 Running time (in s) for processing the incremental tensor of size $100 \times 100 \times t$

HOSVD increase much more slowly than the other algorithms. For the fourth-order tensors, whose slices number is relatively big, compared to R-HOSVD, iT-HOSVD and iST-HOSVD can obtain a speedup of more than $10\times$, and much faster than i-HOSVD; while for the third-order tensors, whose slices number is relatively small, iT-HOSVD and iST-HOSVD do not have so great advantage.

5.2 Time cost evaluation

In the preceding subsection, we only test tensors with $L < J$; see Table 2. To evaluate the time costs shown in Table 1, we generate a random third-order tensor with a large number of slices: $\mathcal{A} \in \mathbb{R}^{100\times 100\times (3\times 10^4)}$. The truncation ranks are set to be (m, m, n), where $m = 10, 30$ and $n = 5, 15, 45$. We choose the first 100 slices as the initialization data. After initialization and setting, we add one slice to a $100 \times 100 \times (t-1)$ tensor, where $t = 101, 102, \ldots, 3 \times 10^4$, and compute the truncated HOSVD of the incremental tensor with the same methods as those of Sect. 5.1. We record the running time of each method for $t = 101, 102, \ldots, 3 \times 10^4$, and show the results in Fig. 5.

As shown in Fig. 5, the running time of iT-HOSVD and iST-HOSVD increases very slowly, and the running time of the other four algorithms increases much faster. Although the incremental algorithm i-HOSVD shows much slower growth of time than the three batch-mode algorithms, it is still not feasible for online computation.

6 Conclusions and future work

By combining the SVD updating with T-HOSVD and ST-HOSVD, we propose two incremental algorithms for truncating the HOSVD: iT-HOSVD, and iST-HOSVD. The complexities and the approximation errors are discussed. As shown in the experiments, the proposed algorithms demonstrate comparable effectiveness with the batch-mode algorithms, while significantly outperform them in terms of efficiency.

The proposed algorithms can be extended to the general case where several slices are appended simultaneously each time. In the future, we will consider the updating and/or downdating algorithm [12] for finding the truncated HOSVD of a tensor by adding and/or deleting a slice from the original tensor.

Declarations

Conflict of interest The authors declared that they have no conflicts of interest to this work.

References

1. Bader, B.W., Kolda, T.G., et al.: MATLAB Tensor Toolbox Version 3.0-dev (2017). https://www.tensortoolbox.org
2. Brand, M.: Fast low-rank modifications of the thin singular value decomposition. Linear Algebra Appl. **415**(1), 20–30 (2006)
3. Bunch, J.R., Nielsen, C.P.: Updating the singular value decomposition. Numer. Math. **31**(2), 111–129 (1978)
4. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. Psychometrika **35**(3), 283–319 (1970)
5. Cheng, Y., Roemer, F., Khatib, O., Haardt, M.: Tensor subspace Tracking via Kronecker structured projections (TeTraKron) for time-varying multidimensional harmonic retrieval. EURASIP J. Adv. Signal Process. **2014**(1), 1–14 (2014)
6. Cocosco, C.A., Kollokian, V., Kwan, R.K.S., Pike, G.B., Evans, A.C.: Brainweb: online interface to a 3D MRI simulated brain database. In: NeuroImage. Citeseer (1997)
7. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
8. De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-(R_1, R_2, ..., R_N) approximation of higher-order tensors. SIAM J. Matrix Anal. Appl. **21**(4), 1324–1342 (2000)
9. De Silva, V., Lim, L.H.: Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM J. Matrix Anal. Appl. **30**(3), 1084–1127 (2008)
10. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010)
11. Gu, M., Eisenstat, S.C.: A stable and fast algorithm for updating the singular value decomposition. Research Report YALEU/DCS/RR-966, Dept. of Computer Science, Yale University (1993)
12. Gu, M., Eisenstat, S.C.: Downdating the singular value decomposition. SIAM J. Matrix Anal. Appl. **16**(3), 793–810 (1995)
13. Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. J. Fourier Anal. Appl. **15**(5), 706–722 (2009)
14. Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics, vol. 16, pp. 1–84. University Microfilms, Ann Arbor, Michigan, No. 10,085 (1970).
15. Hu, W., Li, X., Zhang, X., Shi, X., Maybank, S., Zhang, Z.: Incremental tensor subspace learning and its applications to foreground segmentation and tracking. Int. J. Comput. Vis. **91**(3), 303–327 (2011)

16. Kilmer, M.E., Braman, K., Hao, N., Hoover, R.C.: Third-order tensors as operators on matrices: a theoretical and computational framework with applications in imaging. SIAM J. Matrix Anal. Appl. **34**(1), 148–172 (2013)
17. Kilmer, M.E., Martin, C.D.: Factorization strategies for third-order tensors. Linear Algebra Appl. **435**(3), 641–658 (2011)
18. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
19. Letourneau, P.D., Baskaran, M., Henretty, T., Ezick, J., Lethin, R.: Computationally efficient CP tensor decomposition update framework for emerging component discovery in streaming data. In: 2018 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–8. IEEE (2018)
20. Ma, X., Schonfeld, D., Khokhar, A.: Dynamic updating and downdating matrix SVD and tensor HOSVD for adaptive indexing and retrieval of motion trajectories. In: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1129–1132. IEEE (2009)
21. Minster, R., Saibaba, A.K., Kilmer, M.E.: Randomized algorithms for low-rank tensor decompositions in the Tucker format. SIAM J. Math. Data Sci. **2**(1), 189–215 (2020)
22. Moonen, M., Van Dooren, P., Vandewalle, J.: A singular value decomposition updating algorithm for subspace tracking. SIAM J. Matrix Anal. Appl. **13**(4), 1015–1038 (1992)
23. Nion, D., Sidiropoulos, N.D.: Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. IEEE Trans. Signal Process. **57**(6), 2299–2310 (2009)
24. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
25. Pasricha, R., Gujral, E., Papalexakis, E.E.: Identifying and alleviating concept drift in streaming tensor decomposition. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 327–343. Springer (2018)
26. Rodriguez, P., Wohlberg, B.: Incremental principal component pursuit for video background modeling. J. Math. Imaging Vis. **55**(1), 1–18 (2016)
27. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
28. Seeling, P., Reisslein, M.: Video transport evaluation with H. 264 video traces. IEEE Commun. Surv. Tutor. **14**(4), 1142–1165 (2011)
29. Smith, S., Huang, K., Sidiropoulos, N.D., Karypis, G.: Streaming tensor factorization for infinite data sources. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 81–89. SIAM (2018)
30. Sobral, A., Baker, C.G., Bouwmans, T., Zahzah, E.: Incremental and multi-feature tensor subspace learning applied for background modeling and subtraction. In: International Conference on Image Analysis and Recognition, pp. 94–103 (2014)
31. Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: dynamic tensor analysis. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 374–383. ACM (2006)
32. Sun, J., Tao, D., Papadimitriou, S., Yu, P.S., Faloutsos, C.: Incremental tensor analysis: theory and applications. ACM Trans. Knowl. Discov. Data (TKDD) **2**(3), 11 (2008)
33. Sun, Y., Guo, Y., Luo, C., Tropp, J., Udell, M.: Low-rank Tucker approximation of a tensor from streaming data. SIAM J. Math. Data Sci. **2**(4), 1123–1150 (2020)
34. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
35. Vandecappelle, M., De Lathauwer, L.: Low multilinear rank updating. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers, pp. 437–441. IEEE (2019)
36. Vandecappelle, M., Vervliet, N., De Lathauwer, L.: Nonlinear least squares updating of the canonical polyadic decomposition. In: 2017 25th European Signal Processing Conference (EUSIPCO), pp. 663–667. IEEE (2017)
37. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Sci. Comput. **34**(2), A1027–A1052 (2012)
38. Zeng, C., Ng, M.K.: Incremental CP tensor decomposition by alternating minimization method. SIAM J. Matrix Anal. Appl. **42**(2), 832–858 (2021)
39. Zhou, S., Vinh, N.X., Bailey, J., Jia, Y., Davidson, I.: Accelerating online CP decompositions for higher order tensors. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1375–1384. ACM (2016)